逻辑与神经之间的桥

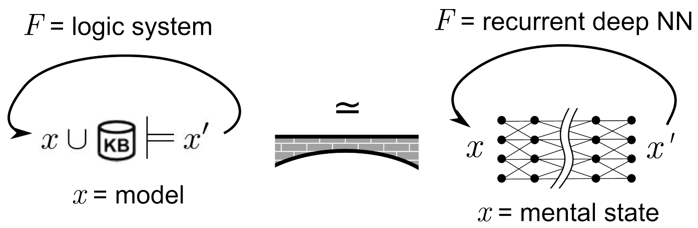
甄景贤 (King-Yin Yan)

General.Intelligence@Gmail.com

Abstract. Logic-based AI 和 connectionist AI 长久分裂，但作者最近发现可以建立两者之间的对应关系。逻辑的结构类似人类的自然语言，但大脑是用神经思考的。机器学习的主要目标，是用 inductive bias 去加快学习速度，但这目标太空泛。将逻辑结构加到神经结构之上，就增加了约束，亦即 inductive bias。

逻辑 AI 那边，「结构」很抽象符号化，但**学习算法**太慢；我的目的是建立一道「桥」，将逻辑 AI 的某部分结构**转移**到神经网络那边。

这个问题搞了很久都未能解决，因为逻辑 AI 那边的结构不是一般常见的数学结构，单是要要表述出来也有很大困难。直到我应用了 model theory 的观点，才找到满意的解决方案：



(1)

首先解释 neural network 那边的结构，然后再解释 logic 那边的结构。

1 神经网络的结构

一个 神经网络 基本上是：

每层的**权重**矩阵　　总层数

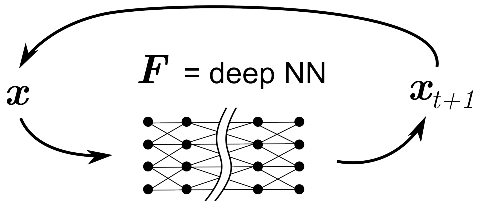
$$F(\boldsymbol{x}) = \sigma(W_1 \sigma(W_2 ... \sigma(W_L\, \boldsymbol{x}))) \tag{2}$$

σ = sigmoid function, applied component-wise （其作用是赋予非线性）

σ 作用在 $\boldsymbol{x}$ 的每个分量上，它的作用在座标变换下**没有不变性**。所以 σ 不是一个向量运算，从而 $\mathbb{X} \ni \boldsymbol{x}$ 的结构也不是**向量空间**的结构。通常习惯把 $\vec{x}$ 写成向量形式，但这有点误导。

如果将神经网络**首尾相接**造成迴路，这是一种智能系统的最简单形式。举例来说，如果要识别「白猫追黑猫」的视像，「猫」这物体要被识别两次，很明显我们不应浪费两个神经网

络去做这件事，所以 迴路 是必须的：



(3)

它的状态方程是：

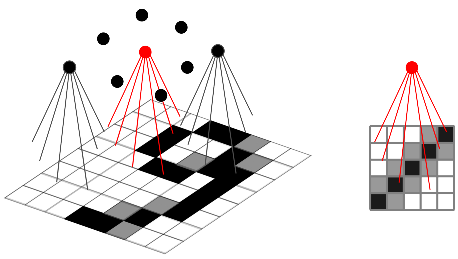
$$\boxed{\text{离散时间}} \qquad \boldsymbol{x}_{t+1} = \boldsymbol{F}(\boldsymbol{x}_t) \tag{4}$$

$$\boxed{\text{连续时间}} \qquad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{5}$$

由此可以看出，$\mathbb{X}$ 是一个**微分流形**。更深入地讲，它是一个力学上的 Hamiltonian 系统，具有 symplectic（辛流形）结构。但这超出了本文範围，详见本篇的姊妹作 [14]。

现在思考一下，神经网络怎样识别模式，或许会有帮助....

考虑最简单的情况，例如提取数字 “9” 的特徵的一层网络。这层网络可以有很多神经元（左图），每个神经元局部地覆盖输入层，即所谓视觉神经元的 local receptive field（右图）。

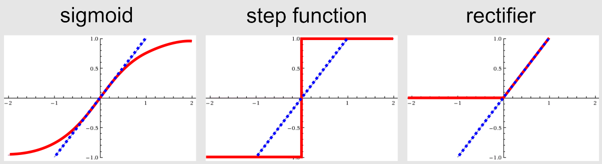


(6)

假设红色的神经元专门负责辨识「对角线」这一特徵。它的方程式是 $\boldsymbol{y} = (W\boldsymbol{x})$。矩阵 W 的作用是 affine「旋转」特徵空间，令我们想要的特徵指向某一方向。然后再用 「**挤压**」想要的特徵和不想要的特徵。Sigmoid 之后的输出，代表某类特徵的**存在与否**，即 $\{0, 1\}$。这是一种资讯的**压缩**。

叉开话题，讲一点 chaos theory：

$^{-1}$ 的作用是「扯」(stretch)，将本来邻近的两点的距离非线性地拉远。看看以下各种常见的激活函数，它们全都是相对於 identity $y = x$ 的非线性 deformation：



(7)

这和 Steven Smale 提出的「马蹄」[10] 非常类似，它是制造混沌的处方之一。Smale 马蹄的另一个变种叫做 baker map，其作用类似於「搓面粉」。换句话说，「拉扯」然后放回原空间，如此不断重复，就会产生混沌 [3] [11]。

这里有一点重大意义：$\boldsymbol{F}^{-1}$ 有混沌的典型特徵：它「对初始状态的微少变化非常敏感」。根据我的观点，神经网络的正向运作是 pattern recogntion = 资讯压缩，反向是反压缩。反向的时候，因为同一概念可以对应於很多不同的输入（$\boldsymbol{F}^{-1}(\boldsymbol{x})$ 可以是很多 patterns），所以输出的微少变化（例如 0.99 和 0.98）会造成 $\boldsymbol{F}^{-1}$ 的极大起伏；换句话说即是有混沌现象。换句话说 $\boldsymbol{F}$ 的逆是 unpredictable 的，简言之，神经网络 $\boldsymbol{F}$ 是**不可逆**的压缩过程。

最近一个有趣的例子是 DeepDream [1]，它用神经网络的 $\boldsymbol{F}^{-1}$ 产生有迷幻感觉的 pre-image，证实了 $\boldsymbol{F}^{-1}$ 可以和原本的 image 差距很大：

(8)

在神经网络的正向运作时似乎没有这种 “stretching”，这似乎不会产生混沌，而且根据 contraction mapping theorem，$\boldsymbol{F}$ 的 iteration 会终止於 fixed point(s)，但前提是 $\boldsymbol{F}$ 是 contractive 的，换句话说 the spectral radius of the Jacobian matrix of $\boldsymbol{F} \leq 1$，但这一点我暂时未能确定（如果对 W 没有限制，这似乎不成立）。

另方面，如果反向是混沌，正向似乎也应该是混沌；在 ergodic theory 里可以计算动态系统的 topological entropy，而这个 entropy 的值是 time-reversal invariant 的。我暂时不清楚如果用别的 entropy 计算会不会不同，有待请教一下 ergodic theory 方面的人....?

总括以上，可以归结出一些原则：

- <u>每个神经元的输出代表某个 feature 的存在与否</u>
- <u>更高层的神经元代表下层 features 之间的**关系**</u>

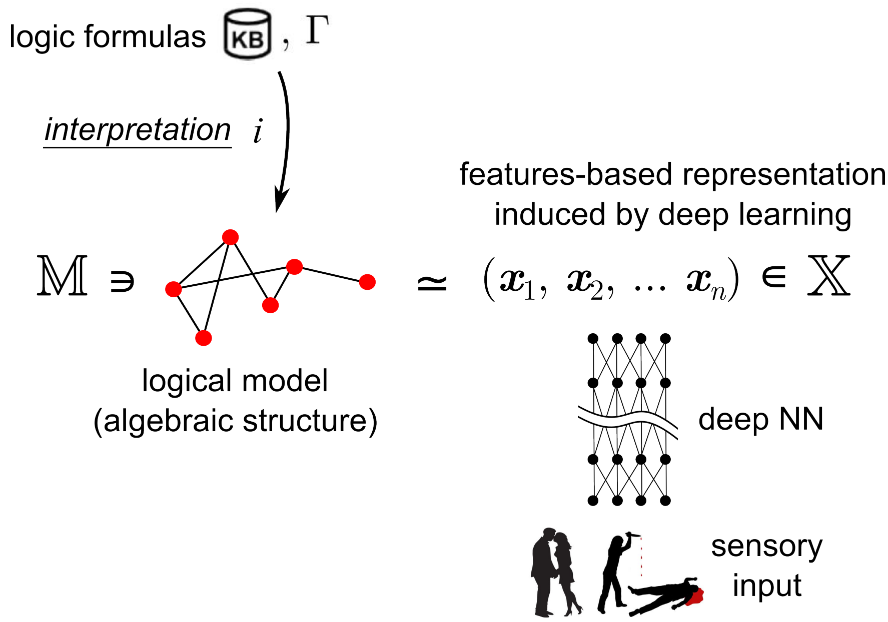
(9)

这些原则暂时未能严格地表述和证明，或者可以叫它做 神经原理 (Neural Postulate)。

凭这个思路推广，可以推测这样的 correspondence：

$$\begin{array}{rccl} & \mathbb{M} & \simeq & \mathbb{X} \\ \text{逻辑物体} & \bullet & \Leftrightarrow & \text{neuron} \\ \text{逻辑关系} & \bullet\!-\!\bullet & \Leftrightarrow & \text{relation between higher and lower neurons} \end{array} \qquad (10)$$

从 model theory 的角度可以这样理解：



(11)

2 逻辑的结构

一个**逻辑系统**可以这样定义：

- 一些 constant symbols，predicate symbols，和 function symbols
- 由上述的原子建立 **命题** (propositions)
- 命题之间可以有连接词：$\neg, \wedge, \vee$ 等
- 建立 **逻辑后果** (consequence) 关系：$\Gamma \vdash \Delta$

逻辑 AI 的 learning algorithm 叫 ILP (inductive logic programming)，这已经是一个 well-established field，如有兴趣可参看我的 ILP tutorial （这版本有点旧，有待更新）[12] 或 de Raedt 的教科书 [8]。ILP 在逻辑式子的符号空间中进行 combinatorial search，缺点是太慢。神经网络的 gradient descent (ie, back-propagation) 快很多，暂时未知是不是要混合这两种算法，还是只用 back-prop....？

我们的目的是将逻辑的结构 transfer 到神经网络。逻辑的结构有两方面：

- 命题的内部结构 (sub-propositional structure)

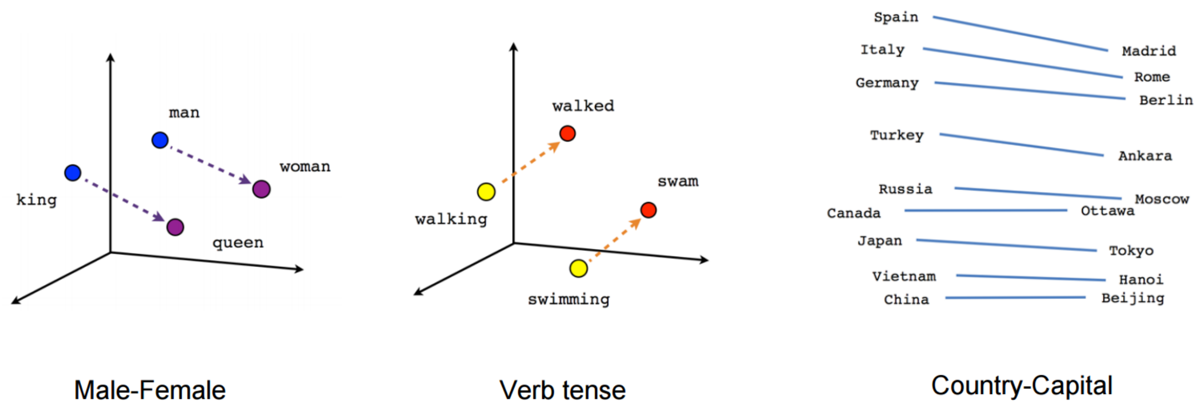
- 多个命题的集合（= 记忆的结构）

以下分述之。谓词逻辑的命题结构比较特别，所以我初时专注於分析它，但后来发现它的影响是次要的；最重要的似乎是《记忆的结构》[13]，这是我们的第 3 篇论文的题目。

2.1 逻辑命题的结构

Semantic distance 与 Compositionality

我们的目的是想将整套逻辑 AI 的器材搬到 连续 的微分流形上去实现。这个做法，是受了 Google 的 Word2Vec [7] 算法的启发，它可以将自然语言的词语 embed 到度量空间中，而且词语之间的距离是 semantic distance （语义学上的距离）：



(12)

当然，word2vec 一经发表之后，很多人开始构思怎样把「句子」也 embed 到度量空间上。第一个想法当然是用 tensor product，例如 “I love you” 这句子就变成了 $I \otimes love \otimes you$。

但这个做法表面上很好，但细思之下却有问题。在语言学 / 逻辑学里有一个重要概念叫 compositionality，它说：

$$\underline{\text{合成物的语义 (semantics) 由其所构成的原子生成，而不依赖於其他资料}} \tag{13}$$

例如「我爱你」由「我、爱、你」三个原子组成。

举个例子，以下这些句子的意义相近：

- *pot calling the kettle black*
- 五十步笑百步
- *Those who live in glass houses should not throw stones*
- 其身不正

虽然可以想像这些句子之间（粗略地）可以建立一一对应，但仍然很难想像用「乘积」乘出来的位置会相近，因为这牵涉到每个词语在句子中的 interpretation ⋆。所以句子的意义不是简单地由词语 bottom-up 生成的。

用 tensor product 的做法，得出的空间结构是 syntactic 而不是 semantic 的，但神经网络学习的重点是 泛化 (generalization)，它基於「邻近的点的意义相近」才能成功，所以一定要 semantic distance。

基於 "features" 的逻辑

上节看到 tensor product 的做法似乎有问题，它用原子 "algebraically" 组合成句子，原子之间的距离可以是语义学相近的，但「乘出来」的句子在语义空间上的位置可能 **很不准确**。另一方面，在神经网络里，「状态」$\boldsymbol{x}$ 是由一组 features 组成的，换句话说，它是一些 features 的 conjunction（也可以看成是乘积）。两者都是乘积，但前者有问题，因为它是由自然语言的字 / 词构成的，而后者是由概念原子 (conceptual atoms / features) 构成的。所以，如果我们避免将自然语言的字 / 句 naïve 地映照到 representation 上，使用概念原子的乘积应该没有问题。

暂时只考虑一个 thought （= 命题）如何表示。

假设思维空间的 metric 是语义的（语义相近则点的距离相近）。根据「神经原理」(9)，每个 thought 必然是由一组 features 表述，例如 $\boldsymbol{x} = (x_1, x_2, ..., x_n)$，每个 x_i 是一个概念原子。

记得逻辑学历史中 George Boole 曾经提出过一种逻辑 ⋆⋆，例如 x 代表「是人的」，y 代表「会死的」，那么「所有人都会死」就是：

$$x\,(1 - y) = 0 \tag{14}$$

这里 x, y 是 "classes"，乘法是集合的 intersection，即 $\cap$。这种乘法和上面描述的句子 / 词语乘法不同；首先，这乘法是 commutative 的，$x \cap y = y \cap x$。如果将一个 thought 表示为：

$$\boldsymbol{x} = x_1, x_2, ..., x_n = x_1 \cap x_2 ... \cap x_n \tag{15}$$

则这些 x_i 可以看成是将所有思维的空间用「二分法」切割，每个 x_i 是一个二分切割。

两个 thoughts 之间的距离可以用 Hamming distance 量度，它是一个 semantic distance，尤其是当 n 比较大时，可能是一个效果不错的 metric。

总括来说：「二分法」逻辑用一组 binary features 去定义一个 thought，一条思维是 n-维 hypercube 上的一个顶点。

⋆ 例如，"pot" 象徵有污点的人。经典 AI 研究者 Jerry Hobbs 开发了自然语言智能系统 Tacitus [5] [4]（这个字有「沉默」的意思），他的理论是基於逻辑上的 abductive interpretations（即逻辑上反方向推导，寻找句子意义的解释），没有这种解释的话，将句子分拆成词语 并不足以组成句子的完整意义；亦即是说，naïve compositionality 在人类语言中并不成立。

⋆⋆ 这和后世为他命名的 Boolean logic 不同。

在经典**谓词逻辑** (predicate logic) 中有一个 “linkage” 的现象，例如以下这个式子（爸爸的爸爸是爷爷）：

$$\forall X\ \forall Y\ \forall Z. \quad \text{grandfather}(X, Z) \leftarrow \text{father}(X, Y) \wedge \text{father}(Y, Z) \tag{16}$$

那意思是说，无论右边的变量（例如 X）怎样改变，左边的 X 必须代入相同的值。这是 代入 (substitution) 的本质。

我们想将 predicate logic 的结构「搬到」神经网络，从抽象意义上说可以有很多种做法，现我讲述一种我设计的比较简单直接的做法：

状态空间的 transition $\boldsymbol{x} \mapsto \boldsymbol{y}$ 看成是逻辑推导 $\boldsymbol{x} \vdash \boldsymbol{y}$，那么 linkage 就是 $\boldsymbol{x}$ 的 i-分量到 $\boldsymbol{y}$ 的 j-分量的 identity map。换句话说，有形如下式的一些 id maps 「埋藏」在 $\boldsymbol{F}$ 里面：

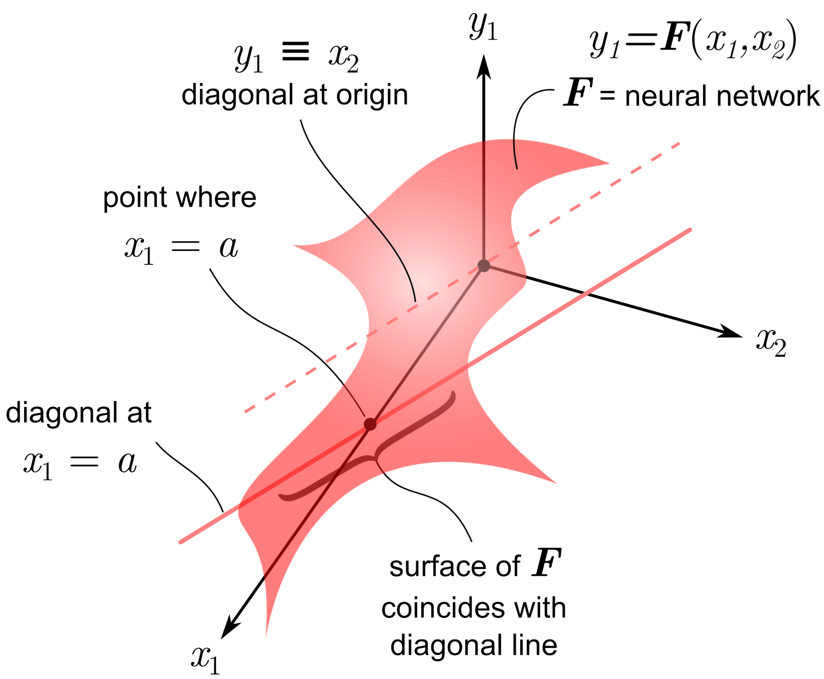
$$\boldsymbol{F}: (x_1, ..., x_i, ..., x_n) \mapsto (y_1, ..., y_j, ..., y_n) \quad (x_i \xrightarrow{id} y_j) \tag{17}$$

注意，我说「埋藏」的意思是：这些 linkage 只在 $\boldsymbol{x}$ 的某些位置才存在（例如，当前提中出现了两个关於「爸爸」的命题时）。更准确地说：

$$\boldsymbol{F}: \begin{cases} y_j \equiv x_i, & \text{if } \boldsymbol{x} = \hat{\boldsymbol{a}} \text{ for some coordinates except } x_i \\ y_h \equiv x_k, & \text{if } \boldsymbol{x} = \hat{\boldsymbol{b}} \text{ for some coordinates except } x_k \\ \text{... etc ...} & \\ \text{is free otherwise} & \end{cases} \tag{18}$$

以上每条等式代表一个 linkage。

一个 linkage 的几何图像如下：



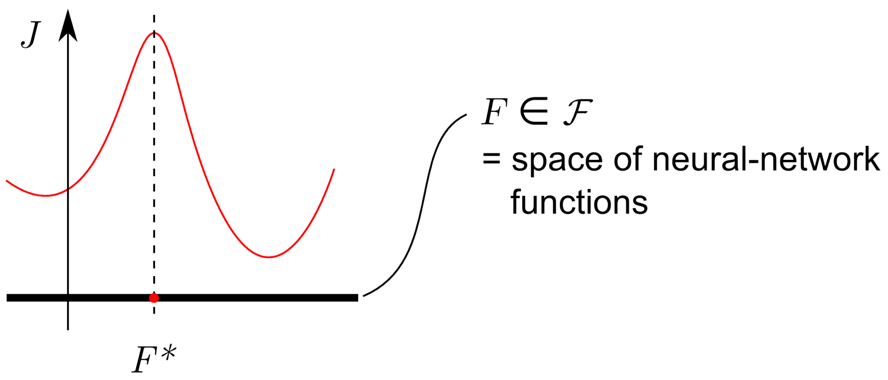
(19)

实际应用中 dim $\boldsymbol{x}$ 可能达到成千上万，其中可以有很多 linkages，很难 visualize。

或者稍为形象化地解释一下： generalization 是智能的一个重要方面，例如由「苏格拉底会死、柏拉图会死」generalize 到「所有人都会死」这样的规律。这种泛化的 **propensity** 是一把双刃剑：有时候可以很聪明，但有时候过度泛化。例如有些男人被女性欺骗后，从此觉得所有女人都说谎。神经网络是一个空间中的 smooth function，它泛化的方法是靠 smoothness：某点的函数值改变时，该点的**邻域**的函数值也随著类似方向改变。神经网络本身没有那种 diagonal 式的泛化倾向（那是逻辑的特徵），但可以模拟它。神经网络 $\boldsymbol{F}$ 对应於逻辑中的 $\vdash$，即是由一些**前题**推出**结论**；$\boldsymbol{F}$ 是一个非常高维数的空间上的一个 hyper-surface。$\boldsymbol{F}$ 里面包含著各种 logic rules，例如在「人」和「死」这些位置上埋藏了「所有人都会死」这个 diagonal；而在「爸爸」和「爷爷」之间埋藏了「爸爸的爸爸是爷爷」这个 diagonal。$\boldsymbol{F}$ 的函数曲面必须经过这些 diagonals。如果没有逻辑的约束，$\boldsymbol{F}$ 的函数曲面就是自由的 smooth function 了。

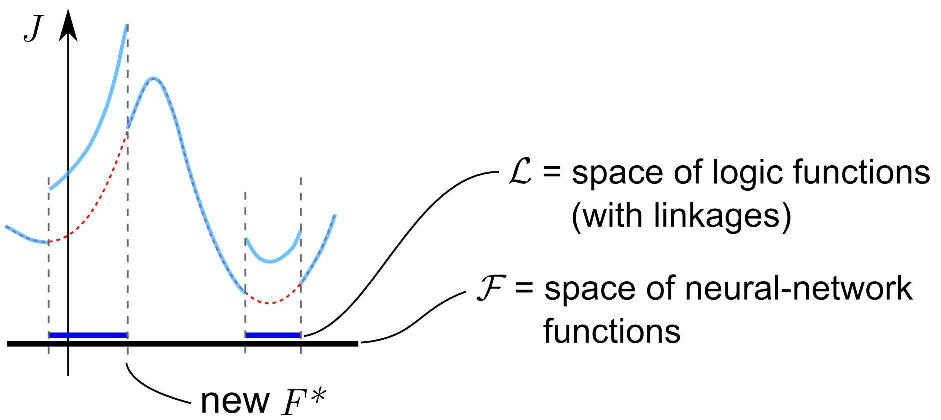
Now，我们关心的是怎样 “combine” 逻辑结构和神经网络结构。在这里我再选择一个特别简单的方法：在 (18) 中的那些等式，构成了一些约束在神经网络的 W 之上的 equational constraints。只需要用这些 constraints 去训练神经网络，就可以实现 linkages。

传统的 back-prop 搜寻 $\boldsymbol{F}$ 的最优值 $\boldsymbol{F}^*$，J 是 score-function，注意底下的 domain 是 泛函空间 (function space)：



(20)

而我们需要的是新的 back-prop，它在「混合」的 function space 上 optimize（至於 exactly 怎样混合我还不清楚），J 在 logic 的 sub-space 上分数较高：



(21)

根据神经网络的 universal approximation theorem [2]，$L > 1$ 层神经网络的函数 $\boldsymbol{F}$ 在函数空间中是 dense 的。换句话说，如果神经元的个数充分多，必然可以学习有任意 linkages 的函数。

如果还有疑惑，可以试试实际上 solve 一个 constraint。为简单起见，弃用  而用 （反正实践中很多深度网络都用 rectifier），这样很方便，因为 rectifier 的前半截就是直线的 identity function。粗略地看，如果只有一层，那 constraint 约束了矩阵 W 的其中一行（有 n 个分量）；每个 linkage 使用 2 个自由度（$\equiv$ 等式使用 1 度，if-then 使用 1 度）。当层数增加时，涉及到的 W 数目递增，而约束等式仍然只是一条，换句话说，应该有很多的自由度可以用。这情况是乐观的。

注意： 暂时我还未设计出使用 linkage 的 learning algorithm，我也不肯定使用了 linkage 之后会不会对学习速度有重大改进....？

用 relation algebra 如何？

以前曾经对 relation algebra [9] [6] 有些憧憬，因为它比较接近人类自然语言，但其实 relation algebra (RA) 和 first-order logic (FOL) 基本上是等效的，在 FOL 里面有 linkage 的复杂性，但在 RA 里面这个复杂性其实也没有消失。可以说「复杂度是守恒的」[***]。

举例来说，在 (16) 中表达「爸爸的爸爸是爷爷」，可以用 RA 更简单地表达：

$$\text{father} \circ \text{father} = \text{grandfather} \tag{22}$$

实际的推导是这样的：

$$\begin{aligned} \text{John father Pete} \\ \text{Pete father Paul} \\ \text{John father} \circ \text{father Paul} \end{aligned} \tag{23}$$

这时要 代入 上面的等式才能得出结论：

$$\text{John grandfather Paul} \tag{24}$$

所以 linkage 的复杂性变成了代数 formula 长度的复杂性。

「长度」本身没有不妥，但我们的目的是将逻辑式子嵌入到状态空间 $\boldsymbol{x}$ 里，这时 variable length 令人很头痛，但 fixed length 没有此问题。

逻辑结构 summary

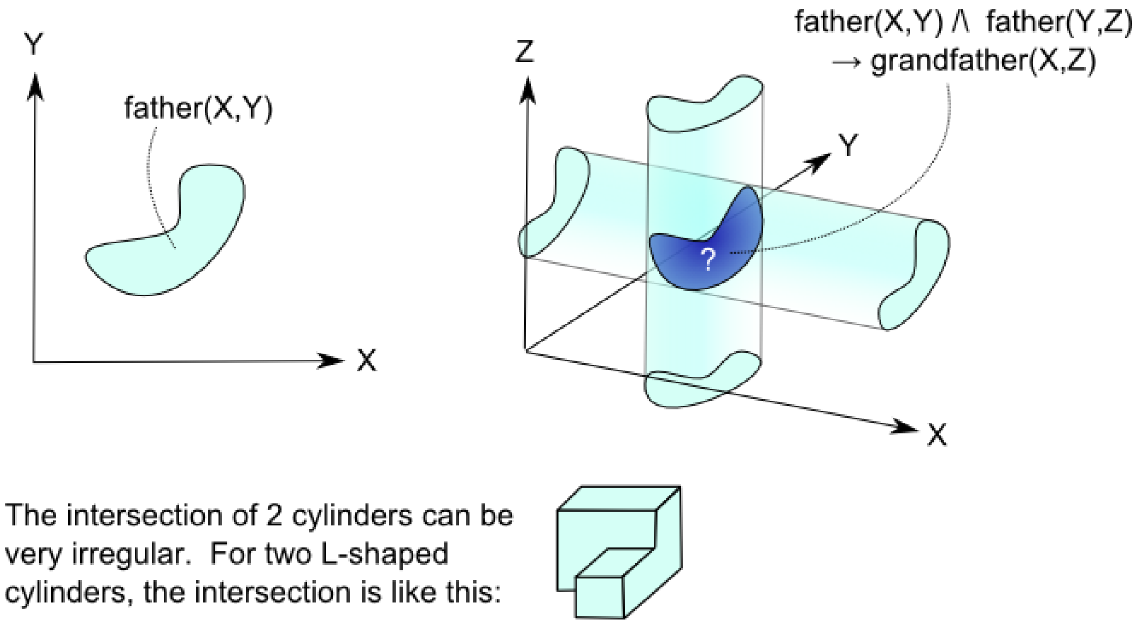
逻辑命题的**内部** (sub-propositional) 结构，传统上有两种方法表达：一是 predicate logic，二是 relation algebra。无论是哪种方式，它的本质都是想处理 variable substitution。有趣的是，「代数」的本质其实也是「代入」，但代数中的代入法则是 implicit 的，所以用代数来 explicitly 表达代入的概念反而很难。

至於 λ-calculus，它本质上就是一种处理 substitution 的方案。而 combinatory logic 和 λ-calculus 是等效的，后者消除了「变量」这概念，但取而代之的是式子的长度更复杂（比 relation algebra 更复杂）。

[***] 这句话是 category theorist Eugenia Cheng 说的。

(First-order) predicate logic 的代数化，可以由 Alfred Tarski 的 cylindric algebra 给出。Higher-order logic (HOL) 的代数化可以用 topoi theory 给出。HOL 等效於 untyped λ-calculus，而 typed λ-calculus (= type theory) 较为弱一点。HOL 这个家族的特点，是 "substitution of equal by equals"，这会令逻辑式子的长度增加。但 predicate logic 的做法是用**实物** (objects) 来做替换，它不会增加式子的长度；但将它代数化的结果是引入了 cylindrification、diagonal set 等概念，例如在 fig. (19) 有 $y_1 \equiv x_2$ 这个 diagonal。

所谓 **cylindrification** 的意思是这样的：



(25)

X, Y, Z 代表 domains，即所有「人」或「物体」(objects) 的集合。物体之间的**关系**是一些 regions（如蓝色那块平面代表 "father" 关系）。两个关系之间的 "and" 就是就是那两个 cylinders 的 intersection，所以叫 cylindric algebra。注意：这里的 (X, Y, Z) 不同於 fig. (19) 的 (x_1, x_2, y_1)，前者是：

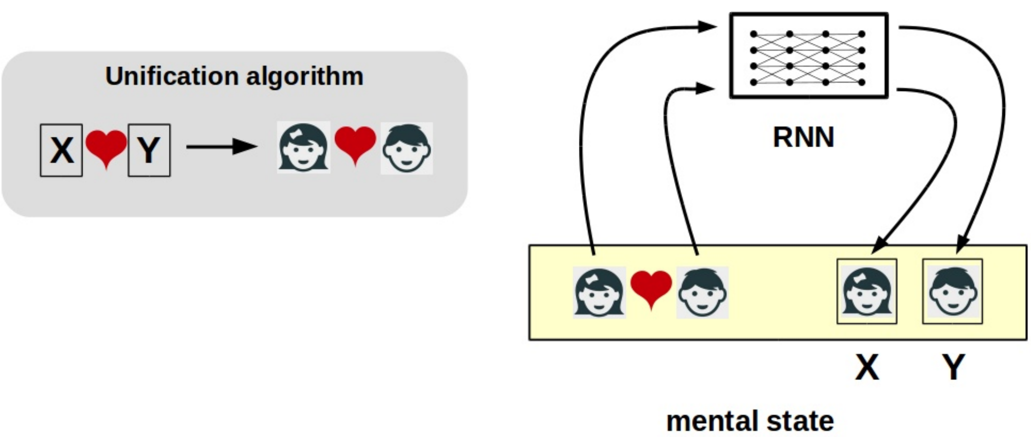
$$\text{father} \wedge \text{father} \tag{26}$$

这个关系，后者是 x_1 = "father" 这个 predicate，而有：

$$\text{father}(x_2, \cdot) \mapsto \text{grandfather}(y_1, \cdot) \tag{27}$$

是个 diagonal (即 linkage: $x_2 \equiv y_1$)。

First-order logic (FOL) 处理 higher-order 关系时不方便，但我选择了 FOL 是因为那 linkage 结构用神经网络似乎较易处理。至於 higher-order relations 则可以用「时间」来搭救（即**分开数个步骤**进行），以下是示意图：



(28)

X，Y 是状态 $\boldsymbol{x}$ 里面的「盒子」，可以充当 variables 的作用。换句话说：适当地利用记忆体，可以将 substitution **分拆**成若干简单的动作，将**空间复杂性**转换成**时间复习性**。人脑的脑电波由 0.5 Hz 到 40 Hz 都有，我们感觉上瞬间的动作可能已经过了若干次迴路。

（这段的 references 太多，我有空再补上....）

Acknowledgement

谢谢 Ben Goertzel（OpenCog 人工智能的创始人）在 AGI mailing list 上和我的讨论。Ben 初次指出神经网络学习和逻辑 inductive 学习的不同，引起我研究两者之间的关系。

References

1. Wikipedia: Deep dreaming (https://en.wikipedia.org/wiki/Deepdreaming).
2. Wikipedia: Universal approximation theorem (https://en.wikipedia.org/wiki/Universal_approximation_theorem).
3. Robert Gilmore and Marc Lefranc. *The topology of chaos: Alice in stretch and squeezeland*. Wiley-VCH, 2011.
4. Hobbs. interpretation as abduction, in book "discourse and inference", Nov 2003.
5. Jerry R Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. Interpretation as abduction. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 95–103. Association for Computational Linguistics, 1988.
6. Roger Maddux. *Relation algebras*. Elsevier, 2006.
7. Mikolov, Sutskever, Chen, Corrado, and Dean. Efficient estimation of word representations in vector space. *Proceedings of workshop at ICLR*, 2013.
8. Luc De Raedt. *Logical and relational learning*. Springer, 2008.
9. Gunther Schmidt. *Relational mathematics*. Cambridge, 2010.
10. Stephen Smale. Differentiable dynamical systems. *Bulletin of the American Mathematical Society*, 1967.
11. Tamás Tél and Márton Gruiz. *Chaotic dynamics: an Introduction based on classical mechanics*. Cambridge, 2006.
12. King Yin Yan. ILP tutorial https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/genifer/Genifer - induction (30 July 2012).pdf.
13. King Yin Yan. The structure of memory. (to be submitted AGI-2017).
14. King Yin Yan, Juan Carlos Kuri Pinto, and Ben Goertzel. Wandering in the labyrinth of thinking – a cognitive architecture combining reinforcement learning and deep learning. (to be submitted AGI-2017).